Executive Summary

This manual summarizes the theory and preliminary verifications of the Preprocessor (PreProc) module, which is part of Kite-FAST (KiteFAST) for airborne wind energy. The theory is largely based on the assumptions made in MBDYN (MBDYN) for its structural dynamics and beam element implementation. Specific assumptions made in KiteFAST, as for example the straight axes for each component of the kite (e.g.,wings, stabilizers, fuselage), are also reflected in how the PreProc handles the stiffness distributions along the various beams.

SOMETHING ABOUT THEORY AND VERIFICATION HERE. The results are encouraging, and future improvements to the code are recommended in this manual.

Acknowledgments

This work was supported by Makani Google X under Contract No. with the National Renewable Energy Laboratory.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Symbols

Acronyms

3D	three-dimensional
AeroDyn	National Renewable Energy Laboratory (NREL)'s aero-hydro-servo-elastic tool for wind turbine design (FAST) aerodynamics module
AeroDyn13	AeroDyn version 13
AeroDyn14	AeroDyn version 14
AeroDyn15	AeroDyn version 15
Controller	Kite-FAST (KiteFAST)'s control system module
FAST8	FAST version 8
InflowWind	KiteFAST's inflow module

KiteAD	Kite-AeroDyn
KiteFAST	Kite-FAST
KiteFASTMBD	MBDYN (MBDYN)'s user module to communicate with KiteFAST
MBDYN	MBDYN
MIP	most important point, fuselage reference point
MoorDyn	KiteFAST's tether module
NREL	National Renewable Energy Laboratory
PreProc	Preprocessor
YAML	human friendly data serialization standard

Greek Symbols

$\Gamma(\theta)$	circulation function of spanwise coordinate θ
$\Gamma(y)$	circulation
Γ_i	i-th circulation value
α_G	geometric angle of attack
α_i	induced angle of attack
α_{inc}	angle of incidence
α	angle of attack
$\hat{\xi}$	nondimensional abscissa along the beam element reference axis equal to -1 at node 1 and to 1 at node 3
$\underline{\Gamma}(y)$	circulation vector
$\underline{\Gamma}_i$	i-th circulation vector

0.1 Overview

The dynamics of semi-rigid kites can be simulated by combining models for aerodynamics and structures. NREL created a new software for the aeroelastic simulation of kites that makes use of MBDYN as the core structural-dynamics solver and a series of modules that account for aerodynamics, tether dynamics, and control system. The software is called KiteFAST. Multiple input files are necessary for the various parts of the software. To create MBDYN's input file, NREL created a special Preprocessor (PreProc).

This manual describes the main aspects of the theory developed and implemented in the PreProc. A detailed commentary of the input file needed for the PreProc is also given.

The approximate way to distribute masses from cross-sectional properties at the first and third node of each element in a beam element is shown below.

0.2 The 3-node Beam Elements and the Preprocessor

The beam elements used in KiteFAST are *beam*3 or 3-node beam elements as described in Masarati 2017. The preprocessor, starting from the end nodes (1 and 3) provided in the input file, will determine the middle node (node 2) location, and two intermediate (Gaussian) points. MBDYN calculates internal forces and moments at the Gaussian points, which are located at $\hat{\xi} = \pm 1/\sqrt{3}$, where $\hat{\xi}$ is the nondimensional abscissa along the beam element reference axis equal to -1 at node 1 and to 1 at node 3.

The stiffness matrices provided at the end nodes *in the kite reference frame* are then transferred to the Gaussian point coordinates, based on a linear interpolation and the abscissa coordinate of the nodes and the Gaussian points.

The user is advised to guarantee sufficient resolution through the end node locations to capture any steep gradients along the various structural elements.

The beams are assumed straight, with reference (major) axis parallel to any one of the kite reference frame axes (x,y, or z). Each component node is therefore located along a constant pair of y,z or x,z or x,y coordinates. At each node, the user can define a twist angle about the beam reference axis that defines the orientation of the cross-section (e.g.,the airfoil pitch).

0.3 The Joints and the Preprocessor

Contiguous beam elements (for example along the starboard wing of the kite) are automatically joined at the common end-nodes. The subcomponents (wings, fuselage, pylons, stabilizers), however, need to be connected among each other unless they can share a physical structural node. For example, in the case of fuselage and either port or starboard wing, the connection can be realized via a 'total joint' (Masarati 2017), which links the degrees of freedom of any two nodes belonging to the two components that must be connected.

In the case of a rotor and a pylon, the connection takes the form of a 'revolute hinge' (Masarati 2017), which allows for a relative rotation along the z-axis of a reference frame common to the two nodes to be connected.

The PreProc sets up all the necessary connections based on the definitions of the various subcomponents, the 'attached component' syntax, and the component keypoint definition (see also Section 0.5).

Note that the attached components are specified following the hierarchy with which they were defined; for example, to specify that the lower port rotor is connected to a pylon node, the syntax in the input file is "rotor_assembly/lower/port/2" or "rotor_assembly/port/lower/2" where the order does not need to be fixed (see Section 0.5).

0.4 Mass Distribution Theory

The *beam3* requires three concentrated masses and inertia at the three nodes. There is no general way to specify masses at the nodes, and the hope is to reduce the error by matching at the end the overall integral (three-dimensional (3D)) inertial properties of the beam element.

In this treatment, we assume that the beam element is rectilinear. For curved beams the theory should be revised.

The main assumption is that the mass and the inertias are modeled as linear distributions. For simplicity's sake, we assume that the coordinate system has the origin at N_A, with the x-axis along the beam element, consistent with MBDYN.

The beam's mass (M) and center of mass coordinates (X_G,Y_G,Z_G) can be calculated as:

$$M = \int_0^{L_b} m(x)dx \tag{1}$$

$$X_G = \frac{\int_0^{L_b} m(x)x_G(x)dx}{M} \tag{2}$$

$$Y_G = \frac{\int_0^{L_b} m(x)y_G(x)dx}{M} \tag{3}$$

$$Z_G = \frac{\int_0^{L_b} m(x)z_G(x)dx}{M} \tag{4}$$

where L_b is the beam's length; $m(x)$ is the cross-sectional mass as a function of the beam's span; $x_G(x)$–$z_G(x)$ are the coordinates as a function of x of the cross-sectional center of mass.

Analogously, we can write the inertial quantities:

$$Ixx = \int_0^{L_b} \left[ixx(x) + m(x)\left(y_G(x)^2 + z_G(x)^2\right)\right] dx \tag{5}$$

$$Iyy = \int_0^{L_b} \left[iyy(x) + m(x)\left(x_G(x)^2 + z_G(x)^2\right)\right] dx \tag{6}$$

$$Izz = \int_0^{L_b} \left[izz(x) + m(x)\left(x_G(x)^2 + y_G(x)^2\right)\right] dx \tag{7}$$

$$Ixy = \int_0^{L_b} \left[ixy(x) + m(x)\left(x_G(x)y_G(x)\right)\right] dx \tag{8}$$

$$Ixz = \int_0^{L_b} \left[ixz(x) + m(x)\left(x_G(x)z_G(x)\right)\right] dx \tag{9}$$

$$Iyz = \int_0^{L_b} \left[iyz(x) + m(x)\left(y_G(x)z_G(x)\right)\right] dx \tag{10}$$

where $ixx(x)$–$izz(x)$ are the cross-sectional second moments of inertia about the centroidal axes as a function of the beam's span x.

Note that because the beam axis is considered along x, $x_G(x) = x$.

The inputs are the quantities $m(x)$, $x_G(x)$–$z_G(x)$, and $ixx(x)$–$izz(x)$ at the end nodes (1st and 3rd node) of the beam. By assuming linear functions of x for all of these quantities, all of the above integral quantities can be analytically

calculated. The expressions of the typical terms are given in Eq. (11):

$$
\begin{aligned}
&m(x) = ax + b \\
&ixx(x)(\text{or} \quad iyy(x), ixy(x), \ldots) = cx + d \\
&y_G(x) = ex + f \\
&z_G(x) = gx + h \\
&m(x)y_G(x) = aex^2 + (af + be)x + bf \\
&y_G(x)z_G(x) = egx^2 + (eh + fg)x + fh \\
&m(x)y_G(x)z_G(x) = aegx^3 + (a(eh + fg) + beg)x^2 + (afh + b(eh + fg))x + bfh \\
&y_G(x)^2 + z_G(x)^2 = (e^2 + g^2)x^2 + 2(ef + gh)x + (f^2 + h^2) \\
&m(x)\left(y_G(x)^2 + z_G(x)^2\right) = a(e^2 + g^2)x^3 + \left(2a(ef + gh) + b(e^2 + g^2)\right)x^2 + \left(a(f^2 + h^2) + 2b(ef + gh)\right)x + b(f^2 + h^2) \\
&M = a\frac{L_b^2}{2} + bL_b \\
&Y_G = ae\frac{L_b^3}{3} + (af + be)\frac{L_b^2}{2} + bfL_b \\
&Ixx = \int_0^{L_b}\left[ixx(x) + m(x)\left(y_G(x)^2 + z_G(x)^2\right)\right]dx = \\
&\quad \left[a(e^2 + g^2)\frac{L_b^4}{4} + \left(2a(ef + gh) + b(e^2 + g^2)\right)\frac{L_b^3}{3} + \left(a(f^2 + h^2) + 2b(ef + gh) + c\right)\frac{L_b^2}{2} + \left(b(f^2 + h^2) + d\right)L_b\right] \\
&Iyy = \int_0^{L_b}\left[cx + d + m(x)\left(x^2 + z_G(x)^2\right)\right]dx = \\
&\quad \left[a(1 + g^2)\frac{L_b^4}{4} + \left(2a(gh) + b(1 + g^2)\right)\frac{L_b^3}{3} + \left(a(h^2) + 2b(gh) + c\right)\frac{L_b^2}{2} + \left(bh^2 + d\right)L_b\right] \\
&Ixy = \int_0^{L_b}\left[ixy(x) + m(x)\left(x_G(x)y_G(x)\right)\right]dx = \int_0^{L_b}\left[cx + d + m(x)\left(xy_G(x)\right)\right]dx = \\
&\quad \int_0^{L_b}\left[aex^3 + (af + be)x^2 + (bf + c)x + d\right]dx = ae\frac{L_b^4}{4} + (af + be)\frac{L_b^3}{3} + (bf + c)\frac{L_b^2}{2} + dL_b \\
&Iyz = \int_0^{L_b}\left[iyz(x) + m(x)\left(y_G(x)z_G(x)\right)\right]dx = \int_0^{L_b}\left[aegx^3 + (a(eh + fg) + beg)x^2 + (afh + b(eh + fg) + c)x + (bfh + d)\right]dx = \\
&\quad aeg\frac{L_b^4}{4} + (a(eh + fg) + beg)\frac{L_b^3}{3} + (afh + b(eh + fg) + c)\frac{L_b^2}{2} + (bfh + d)L_b
\end{aligned}
\tag{11}
$$

where the constants a, b, c, d, e, f, g, h are calculated for each of the parameters from the values at the two nodes as

for example:

$$\begin{aligned} d &= ixx_1 \\ c &= \frac{ixx_3 - ixx_1}{L_b} \end{aligned} \tag{12}$$

where ixx_1 and ixx_3 are the beam's cross-sectional second mass moments of inertia about the centroidal x-axis at first and last node of the beam element, respectively.

The goal is to find three nodal masses and associated inertia tensors that can as-close-as-possible match the same integral quantities. This translates into 27 unknowns: M_i, Y_{Gi}, Z_{Gi}, Ixx_i, Ixx_i, Izz_i, Ixy_i, Ixz_i, and Iyz_i with $i = 1..3$, where: M_i is the i-th lumped mass; y_i, z_i are the x-coordinate of the beam's $i-th$ node and x-coordinate of the beam's $i-th$ node, respectively; Ixx_i–Izz_i and Ixy_i–Iyz_i are the second moments of inertias and the skew moments of inertias associated with the nodal lumped masses.

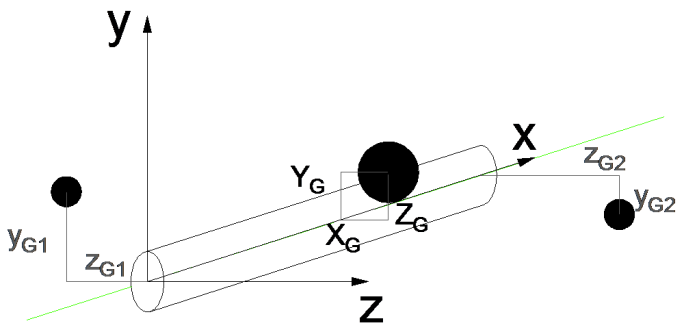

Figure 1. Diagram showing one 2-node subelement and the lumped masses at the nodes and the overall mass lumped at the center of mass of the beam element.

It is convenient to subdivide the 3-node beam element into two 2-node elements (see Fig. 1), and solve the problem for each element, and then sum the inertial quantities at the common node. Eq.s (1)-(5) can be interpreted as the inertial properties of one sub-beam element, if L_b is set to represent the length of that element. In this case, the unknowns reduce to 18: M_i, Y_{Gi}, Z_{Gi}, Ixx_i, Ixx_i, Izz_i, Ixy_i, Ixz_i, and Iyz_i with $i = 1..2$. With this assumption, we begin by writing the mass equalities:

$$M = M_1 + M_2 \tag{13}$$

$$X_G = \frac{M_2 * L_b}{M} \tag{14}$$

$$Y_G = \frac{M_1 * Y_{G1} + M_2 * Y_{G2}}{M} \tag{15}$$

$$Z_G = \frac{M_1 * Z_{G1} + M_2 * Z_{G2}}{M} \tag{16}$$

Then we can write equations for the inertial tensor terms:

$$Ixx = Ixx_1 + M_1\left(Y_{G1}^2 + Z_{G1}^2\right) + Ixx_2 + M_2\left(Y_{G2}^2 + Z_{G2}^2\right) \tag{17}$$

$$Iyy = Iyy_1 + M_1\left(Z_{G1}^2\right) + Iyy_2 + M_2\left(L_b^2 + Z_{G2}^2\right) \tag{18}$$

$$Izz = Izz_1 + M_1\left(Y_{G1}^2\right) + Izz_2 + M_2\left(L_b^2 + Y_{G2}^2\right) \tag{19}$$

$$Ixy = Ixy_1 + Ixy_2 + M_2\left(L_b Y_{G2}\right) \tag{20}$$

$$Ixz = Ixz_1 + Ixz_2 + M_2\left(L_b Z_{G2}\right) \tag{21}$$

$$Iyz = Iyz_1 + M_1\left(Y_{G1} Z_{G1}\right) + Iyz_2 + M_2\left(Y_{G2} Z_{G2}\right) \tag{22}$$

These are 10 equations, and not sufficient to close the problem and solve for the 18 unknowns. One possible approximation is to impose the following 8 additional constraints:

$$Y_{G2} = Y_{G1} \tag{23}$$

$$Z_{G1} = Z_{G2} \tag{24}$$

$$Ixx_1 = Ixx_2 \tag{25}$$

$$Iyy_1 = Iyy_2 \tag{26}$$

$$Izz_1 = Izz_2 \tag{27}$$

$$Ixy_1 = Ixy_2 \tag{28}$$

$$Ixz_1 = Ixz_2 \tag{29}$$

$$Iyz_1 = Iyz_2 \tag{30}$$

By working out some algebra, the equations can be rewritten to give the unknowns:

$$M_2 = \frac{MX_G}{L_b} \tag{31}$$
$$M_1 = M - M_2 \tag{32}$$
$$Y_{G1} = Y_G \tag{33}$$
$$Y_{G2} = Y_G \tag{34}$$
$$Z_{G1} = Z_G \tag{35}$$
$$Z_{G2} = Z_G \tag{36}$$
$$Ixx_1 = \frac{Ixx - M\left(Y_G^2 + Z_G^2\right)}{2} \tag{37}$$
$$Ixx_2 = Ixx_1 \tag{38}$$
$$Iyy_1 = \frac{Iyy - MZ_G^2 - M_2L_b^2}{2} \tag{39}$$
$$Iyy_2 = Iyy_1 \tag{40}$$
$$Izz_1 = \frac{Izz - MY_G^2 - M_2L_b^2}{2} \tag{41}$$
$$Izz_2 = Izz_1 \tag{42}$$
$$Ixy_1 = \frac{Ixy - M_2\left(L_bY_G\right)}{2} \tag{43}$$
$$Ixy_2 = Ixy_1 \tag{44}$$
$$Ixz_1 = \frac{Ixz - M_2\left(L_bZ_G\right)}{2} \tag{45}$$
$$Ixz_2 = Ixz_1 \tag{46}$$
$$Iyz_1 = \frac{Iyz - M\left(Y_GZ_G\right)}{2} \tag{47}$$
$$Iyz_2 = Iyz_1 \tag{48}$$

0.5 Input File

In this section we describe the main keywords and necessary input values for the preprocessor input file. The input file would normally have extension 'yaml' implying that the human friendly data serialization standard (YAML) format is used.

The *title* is used to provide a description of the content the input file refers to. Then a few parameters are set that are directly passed to MBDYN:

- *constants*: this card is used to input constants such as the *gravity* vector components
- *simulation controls*: parameters for MBDYN solver and inputs to KiteFAST are set here:
  - *fast_submodules*: here select *True* or *False* to turn on or off the individual KiteFAST submodules (KiteAeroDyn (KiteAD), KiteFAST's inflow module (InflowWind), KiteFAST's tether module (MoorDyn), KiteFAST's control system module (Controller))
  - *fast_submodule_input_files*: here, the input path and filenames for the KiteFAST submodules are set
  - *time*: here, time step and initial and final time of the simulation are set
    - *initial*: set initial time of the simulation in seconds

        * *timestep*: set simulation time step in seconds
        * *final*: set final time of the simulation in seconds
    - *tolerance*: MBDYN's residual tolerance, i.e.,the tolerance used for the residual test (see Masarati 2017);
    - *max_iterations*: MBDYN's maximum number of iterations for convergence test, (see Masarati 2017));
    - *derivatives*: MBDYN's parameters for the so-called 'derivatives' solution phase, (see Masarati 2017));
        * *tolerance*: set the tolerance on derivatives (a large number if you want to effectively skip the initialization convergence check at the risk of 'rough' transients)
        * *max_iteration*: set the maximum number of iterations for the derivatives phase of the simulation
        * *coefficient*: set the coefficient that relates the state perturbation to the derivative perturbation; generally, the larger the stiffness-to-inertia ratio of the system the smaller the coefficient
    - *linear_solver*: this card is to specify among MBDYN's built-in solvers (e.g.,*naive*), (see Masarati 2017));
    - *rigid_model*: if set to *True*, rigid joints among all the nodes will be created (this feature is not yet enabled);
    - *debug*: if set to *True*, statements will be printed out to standard output from KiteFAST and MBDYN's user module to communicate with KiteFAST (KiteFASTMBD);
    - *ground_weather_station*: set the coordinates of the weather station; this point is passed to KiteFAST to interface with the controller;
        * *location*: weather station x, y, z in global coordinate system
    - *initial_conditions*: set the coordinates and orientation of the kite in global reference frame;
        * *location*: kite's most important point, fuselage reference point (MIP) initial x, y, z in global coordinate system
        * *orientation*: kite's initial roll, pitch, and yaw angles in degrees
        * *velocity*: kite's translational and rotational initial velocities
            · *translational*: kite's translational velocity components in global coordinate system and $\mathrm{m\,s^{-1}}$
            · *rotational*: kite's initial roll, pitch, and yaw rotational velocity in $^\circ\,\mathrm{s^{-1}}$
- *keypoints*: these are x, y, and z values for the subcomponent keypoints, i.e.,the reference points for fuselage, wings, stabilizers etc.. These are in m and in the kite reference frame; the $(0.,0.,0.)$ point (denoted by MIP) is the kite coordinate system origin. The *keypoints* are the respective origins of the reference frames associated with each subcomponent and parallel to the ktie reference frame **jonkman2018**
- *fuselage*: here, the coordinates of the beam element end nodes (minimum of two), and the stiffness matrix and mass distribution for the fuselage are set.
    - *element_end_nodes*: this card includes: x, y, z coordinates of the beam element end-nodes in a reference frame parallel to the kite's main frame and with origin at the fuselage keypoint; the twist angle in $^\circ$ of the cross-section about the component reference axis (either x, y, or z in the kite reference frame); the *attached component*, i.e.,strings identifying any other component that connects to that node; any concentrated (lumped) mass at that node.
    - stiffness_matrix: here, the 21 unique elements of a symmetric stiffness matrix at the coordinates of the previously specified end-nodes and in the kite reference frame. Note that the PreProc will linearly interpolate these values at the actual Gaussian points of each 3-node beam based on their linear distance along the reference axis (see Section 0.2). Units are $\mathrm{N\,m^{-1}}$, $\mathrm{N\,m\,m^{-1}}$, $\mathrm{N\,m\,rad^{-1}}$, $\mathrm{N\,rad^{-1}}$.

  - mass_distribution: provide cross-sectional mass per unit length ($\mathrm{kg\,m^{-1}}$) and coordinates (m) of the center of mass in a plane normal to the reference axis of the component; also specify components of the cross-sectional inertia tensor ($\mathrm{kg\,m}$). The PreProc implements the theory described in Section 0.4 to assign lumped body masses at the beam nodes.

- *wing*: The wing element nodes' coordinates, stiffness matrices, mass distribution, and lumped masses are set in the same way as for the fuselage; however, one more input needed for the wing is the number of control surfaces (flaps) per wing.
  - *number_of_flaps_per_wing*: enter the number of control devices per each wing
  - *starboard*: here, the parameters for the starboard wing are set.
    - *element_end_nodes*: same as for the fuselage, with major axis along kite's positive y axis
    - stiffness_matrix: same as for the fuselage.
    - mass_distribution: same as for the fuselage.
  - *port*: here, the parameters for the port wing are set.
    - *element_end_nodes*: same as for the fuselage, with major axis along kite's negative y axis
    - stiffness_matrix: same as for the fuselage.
    - mass_distribution: same as for the fuselage.

  The stabilizers are treated in a similar fashion:

- *stabilizer*: here, the coordinates of the element nodes, the stiffness matrix and mass distribution, connections to other components, and lumped masses at the nodes for the vertical and horizontal stabilizers are set.
  - *vertical*:
    - *element_end_nodes*: same as for the fuselage, with major axis along kite's negative z axis
    - stiffness_matrix: same as for the fuselage.
    - mass_distribution: same as for the fuselage.
  - *horizontal*:
    - *starboard*:
      - *element_end_nodes*: same as for the starboard wing.
      - stiffness_matrix: same as for the starboard wing.
      - mass_distribution: same as for the starboard wing.
    - *port*:
      - *element_end_nodes*: same as for the port wing.
      - stiffness_matrix: same as for the port wing.
      - mass_distribution: same as for the port wing.

  The pylons are also treated in a similar manner:

- *pylon*: here, the coordinates of the element nodes, the stiffness matrix, mass distribution, lumped masses at the nodes, and connections to other components for the starboard and port pylons are set;
- *starboard*:

  - *1*: This is the inner pylon on the starboard wing.
    - *element_end_nodes*: same as for the vertical stabilizer.
    - stiffness_matrix: same as for the vertical stabilizer.
    - mass_distribution: same as for the vertical stabilizer.
  - *2*: This is the outer pylon on the starboard wing.
    - *element_end_nodes*: same as for the vertical stabilizer.
    - stiffness_matrix: same as for the vertical stabilizer.
    - mass_distribution: same as for the vertical stabilizer.
- *port*:
  - *1*: This is the inner pylon on the port wing.
    - *element_end_nodes*: same as for the vertical stabilizer.
    - stiffness_matrix: same as for the vertical stabilizer.
    - mass_distribution: same as for the vertical stabilizer.
  - *2*: This is the inner pylon on the port wing.
    - *element_end_nodes*: same as for the vertical stabilizer.
    - stiffness_matrix: same as for the vertical stabilizer.
    - mass_distribution: same as for the vertical stabilizer.

Finally, the rotors are specified:

- *rotor_assembly*: here, the mass properties of the rotor and nacelle (mass, inertia tensor and center of mass location), and the initial RPM of the rotor are specified;
- *starboard*:
  - *1*: These are the inner pylon rotor on the starboard wing.
    - *upper*: here, the parameters for the upper rotor are set
      - *rotor*:
        1. initial_rpm: initial rotor speed in RPM
        2. mass_properties: mass (kg), center-of-mass coordinates (m) with respect to a reference frame parallel to the kite's and with origin at the rotor keypoint, and inertia tensor components for the rotor ($\mathrm{kg\,m^2}$).
      - *nacelle*:
        1. mass_properties: mass (kg), center-of-mass coordinates (m) with respect to a reference frame parallel to the kite's and with origin at the rotor keypoint, and inertia tensor components for the nacelle ($\mathrm{kg\,m^2}$).
    - *lower*: here, the parameters for the lower rotor are set
      - *rotor*:
        1. initial_rpm: as for the upper rotor.
        2. mass_properties: as for the upper rotor.

        - *nacelle*:
            1. mass_properties: as for the upper nacelle.
    - *2*: These are the outer pylon rotors on the starboard wing.
        - *upper*: here, the parameters for the upper rotor are set.
            - *rotor*:
                1. initial_rpm: same as inner pylon upper rotor.
                2. mass_properties: same as inner pylon upper rotor.
            - *nacelle*:
                1. mass_properties: same as inner pylon upper nacelle.
        - *lower*: here, the parameters for the lower rotor are set.
            - *rotor*:
                1. initial_rpm: as for the upper rotor.
                2. mass_properties: as for the upper rotor.
            - *nacelle*:
                1. mass_properties: as for the upper nacelle.
- *port*:
    - *1*: These are the inner pylon rotors on the port wing.
        - *upper*: here, the parameters for the upper rotor are set
            - *rotor*:
                1. initial_rpm: initial rotor speed in RPM
                2. mass_properties: mass (kg), center-of-mass coordinates (m) with respect to a reference frame parallel to the kite's and with origin at the rotor keypoint, and inertia tensor components for the rotor ($\mathrm{kg\,m^2}$).
            - *nacelle*:
                1. mass_properties: mass (kg), center-of-mass coordinates (m) with respect to a reference frame parallel to the kite's and with origin at the rotor keypoint, and inertia tensor components for the nacelle ($\mathrm{kg\,m^2}$).
        - *lower*: here, the parameters for the lower rotor are set
            - *rotor*:
                1. initial_rpm: as for the upper rotor.
                2. mass_properties: as for the upper rotor.
            - *nacelle*:
                1. mass_properties: as for the upper nacelle.
    - *2*: These are the outer pylon rotors on the port wing.
        - *upper*: here, the parameters for the upper rotor are set.

        - *rotor*:
            1. initial_rpm: same as inner pylon upper rotor.
            2. mass_properties: same as inner pylon upper rotor.
        - *nacelle*:
            1. mass_properties: same as inner pylon upper nacelle.
    * *lower*: here, the parameters for the lower rotor are set.
        - *rotor*:
            1. initial_rpm: as for the upper rotor.
            2. mass_properties: as for the upper rotor.
        - *nacelle*:
            1. mass_properties: as for the upper nacelle.
- *output*: set the subcomponent (wing, stablizer, etc.) to *true* or *false* to obtain nodal output data from MBDYN.

Bibliography

Masarati, P. 2017. *MBDyn Input File Format — Version 1.7.3*. Tech. rep. Milan, Italy: Politecnico di Milano.